AMENDMENTS

In the Claims

1. (Currently Amended) A method of producing a binary code file comprising:
compiling a plurality of source code instructions; and
outputting a plurality of binary code instructions and compiler annotation, the plurality of binary code instructions being executable by a processor of a computer system, the plurality of binary code instructions are an executable and ~~lining~~ linking format (ELF) binary code file and the compiler annotation is an ELF section.

2. (Original) The method as recited in Claim 1, wherein the compiler annotation enables binary translation to be performed on the plurality of binary code instructions using a non-heuristic approach.

3. (Original) The method as recited in Claim 1, wherein the compiler annotation describes functional characteristics of the plurality of binary code instructions.

4. (Original) The method as recited in Claim 1, wherein the compiler annotation comprises one or more records selected from a module identification (ID), a function ID, a split function ID, a jump table ID, a function pointer initialization ID, a function address assignment ID, an offset expression ID, a data in the text section ID, a volatile load ID, and an untouchable region ID.

5. (Original) The method as recited in Claim 1, wherein the compiling the plurality of source code instructions comprises:
examining the plurality of source code instructions;
reorganizing one or more of the plurality of source code instructions;
translating the plurality of source code instructions into the plurality of binary code instructions;
reorganizing one or more of the plurality of binary code instructions; and

tracking and recording functional characteristics of the plurality of source code instructions and of the plurality of binary code instructions.

6. (Cancelled)

7. (Original) The compiler annotation created by the method of Claim 1.

8. (Currently Amended) A method of translating a source binary code file comprising:

translating a plurality of source binary code instructions utilizing compiler annotation, the plurality of source binary code instructions are an executable and ~~liking~~ linking format (ELF) binary code file and the compiler annotation is an ELF section; and

outputting a plurality of target binary code instructions, the plurality of target binary code instructions being executable by a processor of a computer system.

9. (Original) The method as recited in Claim 8, wherein the compiler annotation enables the translating the plurality of source binary code instructions to be performed on the plurality of source binary code instructions using a non-heuristic approach.

10. (Original) The method as recited in Claim 8, wherein the compiler annotation described functional characteristics of the plurality of binary code instructions.

11. (Original) The method as recited in Claim 8, wherein the compiler annotation comprises one or more records selected from a module identification (ID), a function ID, a split function ID, a jump table ID, a function pointer initialization ID, a function address assignment ID, an offset expression ID, a data in the text section ID, a volatile load ID, and an untouchable region ID.

12. (Original) The method as recited in Claim 8, wherein the translating the plurality of source binary code instructions comprises:

utilizing the compiler annotation to partition the plurality of source binary code instructions into sections, functions and basic blocks; and

building a control-flow graph utilizing the plurality of source binary code instructions and the compiler annotation.

13. (Cancelled)

14. (Original) The method as recited in Claim 8, further comprising:
outputting different compiler annotation.

15. (Original) The plurality of target binary code instructions and the different compiler annotation created by the method of Claim 14.

16. (Currently Amended) A binary code file comprising:
a plurality of binary code instructions, the plurality of binary code instructions being executable by a processor of a computer system; the plurality of binary code instructions are an executable and ~~lining~~ linking format (ELF) binary code file; and
compiler annotation, the compiler annotation being an ELF section;
wherein the compiler annotation enables a binary translator to:
utilize the compiler annotation to partition the plurality of binary code instructions into sections, functions and basic blocks; and
build a control-flow graph utilizing the plurality of binary code instructions and the compiler annotation.

17. (Original) The binary code file as recited in Claim 16, wherein the compiler annotation section enables binary translation to be performed on the plurality of binary code instructions using a non-heuristic approach.

18. (Original) The binary code file as recited in Claim 16, wherein the compiler annotation describes functional characteristics of the plurality of binary code instructions.

19. (Original) The binary code file as recited in Claim 16, wherein the compiler annotation comprises one or more records selected from a module identification (ID), a function

ID, a split function ID, a jump table ID, a function pointer initialization ID, a function address assignment ID, an offset expression ID, a data in the text section ID, a volatile load ID, and an untouchable region ID.

20. (Original) The binary code file as recited in Claim 16, wherein the plurality of binary code instructions and compiler annotation is an ELF format binary code file and the compiler annotation is an ELF section.

21. (Currently Amended) An apparatus for producing a binary code file comprising:

means for compiling a plurality of source code instructions; and

means for outputting a plurality of binary code instructions and compiler annotation, the plurality of binary code instructions are an executable and ~~lining~~ linking format (ELF) binary code file and the compiler annotation is an ELF section.

22. (Original) The apparatus as recited in Claim 21, wherein the compiler annotation enables binary translation to be performed on the plurality of binary code instructions using a non-heuristic approach.

23. (Original) The apparatus as recited in Claim 21, wherein the compiler annotation describes functional characteristics of the plurality of binary code instructions.

24. (Original) The apparatus as recited in Claim 21, wherein the compiler annotation comprises one or more records selected from a module identification (ID), a function ID, a split function ID, a jump table ID, a function pointer initialization ID, a function address assignment ID, an offset expression ID, a data in the text section ID, a volatile load ID, and an untouchable region ID.

25. (Original) The apparatus as recited in Claim 21, wherein the means for compiling the plurality of source code instruction comprises:

means for examining the plurality of source code instructions;

means for reorganizing one or more of the plurality of source code instructions;

means for translating the plurality of source code instructions into the plurality of binary code instructions;

means for reorganizing one or more of the plurality of binary code instructions; and

means for tracking and recording functional characteristics of the plurality of source code instructions and of the plurality of binary code instructions.

26. (Currently Amended) An apparatus for translating a source binary code file comprising:

means for translating a plurality of source binary code instructions utilizing compiler annotation, the plurality of source binary code instructions are an executable and ~~lining~~ linking format (ELF) binary code file and the compiler annotation is an ELF section; and

means for outputting a plurality of target binary code instructions.

27. (Original) The apparatus as recited in Claim 26, wherein the compiler annotation enables the translating the plurality of source binary code instructions to be performed on the plurality of source binary code instructions using a non-heuristic approach.

28. (Original) The apparatus as recited in Claim 26, wherein the compiler annotation describes functional characteristics of the plurality of binary code instructions.

29. (Original) The apparatus as recited in Claim 26, wherein the compiler annotation comprises one or more records selected from a module identification (ID), a function ID, a split function ID, a jump table ID, a function pointer initialization ID, a function address assignment ID, an offset expression ID, a data in the text section ID, a volatile load ID, and an untouchable region ID.

30. (Original) The apparatus as recited in Claim 26, wherein the means for translating the plurality of source binary code instructions comprises:

means for utilizing the compiler annotation to partition the plurality of source binary code instructions into sections, functions and basic blocks; and

means for building a control-flow graph utilizing the plurality of source binary code instructions and the compiler annotation.

31. (Currently Amended) An apparatus for producing a binary code file comprising:
a computer readable medium; and
instructions stored on the computer readable medium to:
compile a plurality of source code instructions; and
output a plurality of binary code instructions and compiler annotation, the plurality of binary code instructions are an executable and ~~lining~~ linking format (ELF) binary code file and the compiler annotation is an ELF section.

32. (Original) The apparatus as recited in Claim 31, wherein the compiler annotation enables binary translation to be performed on the plurality of binary code instructions using a non-heuristic approach.

33. (Original) The apparatus as recited in Claim 31, wherein the compiler annotation describes functional characteristics of the plurality of binary code instructions.

34. (Original) The apparatus as recited in Claim 31, wherein the compiler annotation comprises one or more records selected from a module identification (ID), a function ID, a split function ID, a jump table ID, a function pointer initialization ID, a function address assignment ID, an offset expression ID, a data in the text section ID, a volatile load ID, and an untouchable region ID.

35. (Original) The apparatus as recited in Claim 31, wherein the instructions to compile the plurality of source code instructions comprises instructions to:
examine the plurality of source code instructions;
reorganize one or more of the plurality of source code instructions;
translate the plurality of source code instructions into the plurality of binary code instructions;
reorganize one or more of the plurality of binary code instructions; and

track and record functional characteristics of the plurality of source code instructions and of the plurality of binary code instructions.

36. (Currently Amended) An apparatus for translating a source binary code file comprising:

a computer readable medium; and

instructions stored on the computer readable medium to:

translate a plurality of source binary code instructions utilizing compiler annotation, the plurality of source binary code instructions are an executable and ~~lining~~ linking format (ELF) binary code file and the compiler annotation is an ELF section; and

output a plurality of target binary code instructions.

37. (Original) The apparatus as recited in Claim 36, wherein the compiler annotation enables the translating the plurality of source binary code instructions to be performed on the plurality of source binary code instructions using a non-heuristic approach.

38. (Original) The apparatus as recited in Claim 36, wherein the compiler annotation describes functional characteristics of the plurality of binary code instructions.

39. (Original) The apparatus as recited in Claim 36, wherein the compiler annotation comprises one or more records selected from a module identification (ID), a function ID, a split function ID, a jump table ID, a function pointer initialization ID, a function address assignment ID, an offset expression ID, a data in the text section ID, a volatile load ID, and an untouchable region ID.

40. (Original) The apparatus as recited in Claim 36, wherein the instructions to translate the plurality of source binary code instructions comprises instructions to:

utilize the compiler annotation to partition the plurality of source binary code instructions into sections, functions and basic blocks; and

build a control-flow graph utilizing the plurality of source binary code instructions and the compiler annotation.